Journal of Statistical Software

June 2007, Volume 19, Issue 6. http://www.jstatsoft.org/

Rational Arithmetic **Mathematica** Functions to Evaluate the One-sided One-sample K-S Cumulative Sampling Distribution

J. Randall Brown
Kent State University

Milton E. Harvey
Kent State University

Abstract

One of the most widely used goodness-of-fit tests is the Kolmogorov-Smirnov (K-S) family of tests which have been implemented by many computer statistical software packages. To calculate a p value (evaluate the cumulative sampling distribution), these packages use various methods including recursion formulae, limiting distributions, and approximations of unknown accuracy developed over thirty years ago. Based on an extensive literature search for the one-sided one-sample K-S test, this paper identifies two direct formulae and five recursion formulae that can be used to calculate a p value and then develops two additional direct formulae and four iterative versions of the direct formulae for a total of thirteen formulae. To ensure accurate calculation by avoiding catastrophic cancelation and eliminating rounding error, each formula is implemented in rational arithmetic. Linear search is used to calculate the inverse of the cumulative sampling distribution (find the confidence interval bandwidth). Extensive tables of bandwidths are presented for sample sizes up to $2,000$. The results confirm the hypothesis that as the number of digits in the numerator and denominator integers of the rational number test statistic increases, the computation time also increases. In comparing the computational times of the thirteen formulae, the direct formulae are slightly faster than their iterative versions and much faster than all the recursion formulae. Computational times for the fastest formula are given for sample sizes up to fifty thousand.

Keywords: K-S sampling distributions, K-S one-sided one-sample probabilities, K-S confidence bands, rational arithmetic.

1. Introduction

The Kolmogorov-Smirnov (K-S) family of tests is one of the most widely used goodness-of-fit tests and is included in many nonparametric statistics texts (see recent texts Gibbons

and Chakraborti (2003), Sprent and Smeeton (2001), Conover (1999), Daniel (1990)). These include the one-sided one-sample, two-sided one-sample, one-sided two-sample, and the two-sided two-sample tests. The K-S family of tests also include restricted range tests (comparing distributions over a portion of their range) and ratio tests (the ratio of one distribution to another).

For sample size n, the most common K-S test is the two-sided one-sample test which uses the maximum absolute distance D_n between the hypothesized continuous cumulative distribution $F(x)$ and the empirical cumulative distribution $F_n(x)$, $D_n = \sup_{-\infty<x<\infty} |F_n(x) - F(x)|$, as the random variable. In a hypothesis testing application, computing the test statistic d is relatively easier than evaluating the cumulative sampling distribution to determine the p value, $P[D_n \geq d]$. The cumulative sampling distribution is a piecewise polynomial that is different for each sample size n and whose complexity rapidly grows with increasing n so that it has not even been generated let alone used for $n > 31$ (see Ruben and Gambino (1982) and Drew, Glen, and Leemis (2000)). Consequently, the limiting distribution, various recursion formulae, and various approximations have been used to evaluate the cumulative sampling distribution. In addition, many computer statistical software packages such as SPSS, STATISTICA, R, Numerical Recipes, and **IMSL** include K-S tests. Although a recursion formula will theoretically determine the p value $P[D_n \geq d]$ for a particular value d of the test statistic, the complexity of the formula is such that roundoff error and catastrophic cancelation can greatly reduce the accuracy of the calculations. Since most procedures used today were developed on pre-1978 computers where only machine precision was available, the accuracy of their results is not known exactly. Consequently, recursion formulae have only been used to generate tables for sample sizes of $n \leq 40$ and various approximations of unknown accuracy have been used for $n > 40$. Using recursion formula and rational arithmetic, Brown and Harvey (2005) were able to compute p values for sample sizes up to two thousand, $n = 2,000$.

In addition to the two-sided one-sample case (absolute difference between hypothesized and empirical), the one-sided one-sample (difference between hypothesized and empirical) cumulative sampling distribution is a complex series that can also be evaluated by recursion formulae. Indeed, many computer statistical software packages that implement both the two-sided and one-sided one-sample K-S test use different methods to calculate the p values. Table 1 summarizes the strategies used by some of these commercial statistical packages to calculate two-sided and one-sided one-sample p values. Although these packages compute the K-S test statistic in the same way, there is considerable difference in the way they evaluate the cumulative sampling distribution (calculate p values). The Numerical Recipes statistical subroutines in Press, Teukolsky, Vetterling, and Flannery (1992) use an approximation by Stephens (1970) to generate p values for the two-sided one-sample K-S cumulative sampling distribution. SPSS 15.0 (SPSS Inc. 2006) does not state how the p value for the two-sided one-sample K-S test is calculated. However, in 2002, the manual for SPSS 11.0 stated that the statistical software package used a modification of the limiting distribution derived by Feller (1948) and used by Smirnov (1948). Assuming SPSS has not changed how the p value is calculated, the 2002 method is listed in Table 1. The STATISTICA (StatSoft, Inc. 2006) software package uses the critical values tabulated by Massey (1950) and Massey (1951) to generate their p values. **IMSL** (Visual Numerics 2006) computes both the one-sided and two-sided K-S test statistics and then gives p values for each test. Specifically, for the one-sided K-S test and sample sizes $n \leq 80$, **IMSL** uses a recursion formula in Conover (1972) to compute the exact p values for the one-sided sampling distribution, but for large sample sizes

Statistical Software Package	Type of One-sample K-S Test	Used To Calculate p values
IMSL	One-sided	For $n \leq 80$, recursion formula by Conover (1972).
		For $n > 80$, limiting distribution derived by Feller (1948).
	Two-sided	Double the corresponding one-sided p value.
Numerical Recipes	Two-sided	Approximation by Stephens (1970).
R	One-sided	For $n \leq 100$, direct formula by Smirnov (1944).
		For $n > 100$, limiting distribution derived by Feller (1948).
	Two-sided	For $n \leq 100$, program by Marsaglia *et al.* (2003).
		For $n > 100$, limiting distribution derived by Kolmogorov (1933).
SPSS	Two-sided	Modification of limiting distribution derived by Feller (1948).
STATISTICA	Two-sided	Critical values tabulated by Massey (1950) and Massey (1951).

Table 1: Statistical software packages and the one-sample K-S test.

$n > 80$, it uses the one-sided limiting distribution. **IMSL** then doubles the one-sided p values to get the corresponding two-sided p values. Like **IMSL**, the R statistical software package computes both the one-sided and two-sided K-S test statistics but uses different methods to compute the p values. For large sample sizes $n > 100$, R Development Core Team (2006) states that the asymptotic distributions are used (presumably the formula by Feller (1948) for the one-sided K-S test and the formula by Kolmogorov (1933) for the two-sided case). For small sample sizes $n \leq 100$, the one-sided K-S test uses the direct formula by Smirnov (1944) to calculate the p value and the two-sided K-S test uses the matrix formula of Durbin (1973) as implemented by Marsaglia, Tsang, and Wang (2003). In 2002, the R statistical software package instead computed the one-sided p values using the techniques in Conover (1972) and doubled the one-sided p value to get the two-sided p value. It is not clear in 2002 whether R also used the one-sided limiting distribution for large sample sizes.

In addition to the one-sample K-S cumulative sampling distributions, there are sampling distributions for many other K-S type statistics including two-sample (difference between two empirical distributions), restricted range (two distributions compared over a portion of their range), and ratios (the ratio of one distribution to another). For the two-sample K-S case with sample sizes m and n, the one-sided cumulative sampling distribution is a simple formula for $m = n$ and a complex series for $m \neq n$. The two-sided two-sample K-S cumulative sampling distribution is a complex series for $m = n$ while a recursion formula is needed for $m \neq n$. There are many K-S one-sided one-sample restricted range and ratio cumulative sampling distributions whose formulae are known but are very complex expressions.

Given the advances in computing power and computational software in the past thirty years, it is time to reconsider the entire area and if possible, devise techniques to accurately and

quickly evaluate K-S cumulative sampling distributions. Since the entire K-S cumulative sampling distribution area is so large, the question is where we should begin a comprehensive evaluation of the alternate formulae for the various K-S cumulative sampling distributions. Because of the large number and complexity of the formulae, the one-sided one-sample K-S restricted range and ratio areas are not good starting points. Similarly, since two-sample sizes require more computational work than one-sample size, the two-sample area should be done after the one-sample area. The one-sample area contains two tests, the two-sided one-sample K-S test and the one-sided one-sample K-S test. Since Brown and Harvey (2005) have already investigated the two-sided one-sample case, this paper will investigate the one-sided one-sample case.

This paper reviews the one-sided one-sample K-S cumulative sampling distribution formulae, devises rational arithmetic implementations of each formula, verifies the validity of each implementation by determining if each implementation gets exactly the same p value over a broad range of examples, develops an efficient method to calculate the bandwidth (the inverse of the cumulative sampling distribution), and finally compares the computational times needed for each implementation to determine the fastest formula.

2. One-sided one-sample K-S sampling distribution formulae

There are two one-sided one-sample random variables: the one-sided upper random variable $D_n^+ = \sup_{-\infty<x<\infty}\{F_n(x) - F(x)\}$ and the one-sided lower random variable $D_n^- = \sup_{-\infty<x<\infty}\{F(x) - F_n(x)\}$. Since by symmetry D_n^+ and D_n^- have the same cumulative sampling distribution, D_n^+ is used to represent both cases.

Based on an extensive literature search for the one-sided one-sample K-S test, this paper identifies two direct formulae and five recursion formulae that can be used to calculate a p value, $P[D_n^+ \geq d^+]$, and then develops two additional direct formulae and four iterative versions of the direct formulae for a total of thirteen formulae. Table 2 contains a summary of the thirteen formulae which are developed in this section.

2.1. Direct formulae

A closed form expression of the one-sided one-sample K-S cumulative sampling distribution was developed by Smirnov (1944) and verified by many scholars including Feller (1948) and Birnbaum and Tingey (1951). For $0 < d^+ \leq 1$ and sample size n, Smirnov's formula denoted by SmirnovD in this paper is shown in the first row of Table 3 where $\lfloor n(1-d^+)\rfloor$ is the greatest integer less than or equal to $n(1-d^+)$. Dwass (1959) derived a different formula denoted by DwassD that is also shown in Table 3. Both the SmirnovD and DwassD formulae are also derived in Durbin (1973). A second form of the Smirnov distribution denoted by SmirnovAltD and a second form of the Dwass distribution denoted by DwassAltD are derived by factoring $1/n^{n-1}$ out of their respective formulae. The alternate forms, SmirnovAltD and DwassAltD, shown in Table 3 may be faster than the original formulations because the terms inside the summation are simpler rational numbers.

In most applications, the test statistic d^+ is less than 0.5 and usually much less than 0.5 which means the number of terms $\lfloor n(1-d^+)\rfloor + 1$ in the SmirnovD and SmirnovAltD formulae can be close to the sample size n. In comparison, the number of terms $\lfloor nd\rfloor + 1$ in the DwassD and DwassAltD formulae is much less than the number in the SmirnovD and SmirnovAltD

Reference	Type Formula	Formula Name
Smirnov (1944)	Direct	SmirnovD
	Direct	SmirnovAltD
	Iterative	SmirnovI
	Iterative	SmirnovAltI
Dwass (1959)	Direct	DwassD
	Direct	DwassAltD
	Iterative	DwassI
	Iterative	DwassAltI
Daniels (1945)	Recursion	Daniels
Noe and Vandewiele (1968)	Recursion	Noe
Steck (1969)	Recursion	Steck
Conover (1972)	Recursion	Conover
Kotelnikov and Chmaladze (1983)	Recursion	Bolshev

Table 2: Thirteen formulae to calculate a K-S one-sided one-sample p value

formulae and should therefore take less computation time. On the other hand, all the terms in the SmirnovD and SmirnovAltD formulae are positive while the terms in the DwassD and DwassAltD formulae alternate signs. This means that the Dwass formulae are much more susceptible to error than the Smirnov formulae. Note that implementing the formulae in rational arithmetic removes the issue of computational error as all computations are exact. This will be discussed in detail in Section 3.1.

2.2. Iterative formulae

Each of the four formulae in Table 3 can be transformed into an iterative formula which might be faster than the original formula. The Smirnov formula will be used to illustrate the process. Let γ_j be the value of the jth term in the series and let x_j be the iterative factor that converts γ_{j-1} to γ_j so that $\gamma_j = x_j\gamma_{j-1}$. The following derives x_j for the SmirnovD and SmirnovAltD formula. Note that the x_j must be the same for the SmirnovD and SmirnoAltD formulae since the SmirnovAltD formula is simply the SmirnovD formula with $1/n^{n-1}$ factored out.

$$\begin{aligned}
\gamma_j &= d^+\binom{n}{j}\left(1-d^+-\frac{j}{n}\right)^{n-j}\left(d^+ + \frac{j}{n}\right)^{j-1} \\
\gamma_{j-1} &= d^+\binom{n}{j-1}\left(1-d^+-\frac{j-1}{n}\right)^{n-j+1}\left(d^+ + \frac{j-1}{n}\right)^{j-2} \\
x_j &= \frac{\gamma_j}{\gamma_{j-1}} = \frac{d^+\binom{n}{j}\left(1-d^+-\frac{j}{n}\right)^{n-j}\left(d^+ + \frac{j}{n}\right)^{j-1}}{d^+\binom{n}{j-1}\left(1-d^+-\frac{j-1}{n}\right)^{n-j+1}\left(d^+ + \frac{j-1}{n}\right)^{j-2}}
\end{aligned}$$

Type	Formula to Compute $P\left[D_n^+ \geq d^+\right]$ for $0 < d^+ \leq 1$
SmirnovD	$d^+ \sum_{j=0}^{\lfloor n(1-d^+)\rfloor} \binom{n}{j} \left(1 - \frac{j}{n} - d^+\right)^{n-j} \left(\frac{j}{n} + d^+\right)^{j-1}$
DwassD	$1 - d^+ \sum_{j=0}^{\lfloor nd^+\rfloor} \binom{n}{j} \left(1 - \frac{j}{n} + d^+\right)^{n-j-1} \left(\frac{j}{n} - d^+\right)^{j}$
SmirnovAltD	$\frac{d^+}{n^{n-1}} \sum_{j=0}^{\lfloor n(1-d^+)\rfloor} \binom{n}{j} \left(n - d^+n - j\right)^{n-j} \left(d^+n + j\right)^{j-1}$
DwassAltD	$1 - \frac{d^+}{n^{n-1}} \sum_{j=0}^{\lfloor nd^+\rfloor} \binom{n}{j} \left(n - j + d^+n\right)^{n-j-1} \left(j - d^+n\right)^{j}$

$\lfloor n(1-d^+)\rfloor$ is the greatest integer less than or equal to $n(1-d^+)$

Table 3: K-S one-sided one-sample direct formulae.

$$= \frac{(n-j+1)\left(d^+n+j\right)}{j\left(n-d^+n-j\right)} \times \left[1 - \frac{1}{n-d^+n-j+1}\right]^{n-j+1} \times \left[1 + \frac{1}{d^+n+j-1}\right]^{j-2}$$

For the DwassD and DwassAltD formulae, let y_j be the iterative factor that converts γ_{j-1} to γ_j so that $\gamma_j = y_j\gamma_{j-1}$. The following derives y_j for the DwassD and DwassAltD formulae.

$$\begin{aligned}
\gamma_j &= d^+\binom{n}{j}\left(1 - \frac{j}{n} + d^+\right)^{n-j-1}\left(\frac{j}{n} - d^+\right)^{j} \\
\gamma_{j-1} &= d^+\binom{n}{j-1}\left(1 - \frac{j-1}{n} + d^+\right)^{n-j}\left(\frac{j-1}{n} - d^+\right)^{j-1} \\
x_j &= \frac{\gamma_j}{\gamma_{j-1}} = \frac{d^+\binom{n}{j}\left(1 - \frac{j}{n} + d^+\right)^{n-j-1}\left(\frac{j}{n} - d^+\right)^{j}}{d^+\binom{n}{j-1}\left(1 - \frac{j-1}{n} + d^+\right)^{n-j}\left(\frac{j-1}{n} - d^+\right)^{j-1}} \\
&= \frac{(n-j+1)\left(j-d^+n\right)}{j\left(n-j+1+d^+n\right)} \times \left[1 - \frac{1}{n-j+1+d^+n}\right]^{n-j-1} \times \left[1 + \frac{1}{j-1-d^+n}\right]^{j-1}
\end{aligned}$$

Using the same process as above, the iterative formula for SmirnovD denoted by SmirnovI can be derived. Similarly, the iterative formulae for SmirnovAltD, DwassD, and DwassAltD can be derived and are denoted by SmirnovAltI, DwassI, and DwassAltI respectively. The results are shown in Table 4.

Type	Iterative Formula
Smirnov	$x_j = \frac{(n-j+1)\,(d^+n+j)}{j\,(n-d^+n-j)} \times \left[1 - \frac{1}{n-d^+n-j+1}\right]^{n-j+1} \times \left[1 + \frac{1}{d^+n+j-1}\right]^{j-2}$
Dwass	$y_j = \frac{(n-j+1)(j-d^+n)}{j(n-j+1+d^+n)} \times \left[1 - \frac{1}{n-j+1+d^+n}\right]^{n-j-1} \times \left[1 + \frac{1}{j-1-d^+n}\right]^{j-1}$

Name	Initial Value	Iteration	$P\left[D_n^+ \geq d^+\right]$
SmirnovI	$\gamma_0 = (1-d^+)^n$	$\gamma_j = x_j\gamma_{j-1}$	$\sum_{j=0}^{\lfloor n(1-d^+)\rfloor} \gamma_j$
SmirnovAltI	$\gamma_0 = n^n(1-d^+)^n$	$\gamma_j = x_j\gamma_{j-1}$	$n^n \sum_{j=0}^{\lfloor n(1-d^+)\rfloor} \gamma_j$
DwassI	$\gamma_0 = d^+(1+d^+)^{n-1}$	$\gamma_j = y_j\gamma_{j-1}$	$1 - \sum_{j=0}^{\lfloor nd^+\rfloor} \gamma_j$
DwassAltI	$\gamma_0 = d^+(n+d^+n)^{n-1}$	$\gamma_j = y_j\gamma_{j-1}$	$1 - \left[\left(\sum_{j=0}^{\lfloor nd^+\rfloor} \gamma_j\right)/n^{n-1}\right]$

$\lfloor n(1-d^+)\rfloor$ is the greatest integer less than or equal to $n(1-d^+)$

Table 4: K-S one-sided one-sample iterative formulae.

In addition to the four direct formulae and four iterative formulae, five recursion formulae to compute the one-sided one-sample K-S p value have been derived and are presented in chronological order in the next five subsections.

2.3. Daniels' recursion formula

Daniels (1945) derived a difference equation that was later restated by Noe and Vandewiele (1968). The form of Daniels' recursion formula (referred to henceforth as Daniels) shown below is derived by solving the difference equation for $Q_i(1)$. The recursion formulae use the test statistic $d^+ = t/n$ or $t = d^+n$.

$$\begin{aligned} Q_0(1) &= 1 \\ Q_i(1) &= -\sum_{k=0}^{i-1} \binom{i}{k} Q_k(1) \left[\max\left(\frac{i-t}{n}, 0\right) - 1\right]^{i-k} \qquad \text{for } i = 1, 2, \cdots, n \\ P\left(D_n^+ \geq \frac{t}{n}\right) &= 1 - Q_n(1) \end{aligned}$$

2.4. Noe and Vandewiele recursion formula

Since the Daniels recursion formula has both positive and negative terms, Noe and Vandewiele (1968) derived an alternate recursion formula that has only non-negative terms. Noe (1972) later added a correction to this recursion formula. The particular form of the recursion formula (referred to henceforth as Noe) listed below containing Noe's correction is taken from Shorack and Wellner (1986), page 363, formulas (24) through (28).

$$\begin{aligned} Q_0(0) &= 1 \\ Q_m(m) &= 0 \qquad \text{for } 1 \le m \le n+1 \\ Q_i(m) &= \sum_{k=0}^{i} \binom{i}{k} Q_k(m-1) \left[\max\left(\frac{m-t}{n}, 0\right) - \max\left(\frac{m-t-1}{n}, 0\right)\right]^{i-k} \\ &\qquad \text{for } 0 \le i \le m-1, 1 \le m \le n+1 \\ P\left(D_n^+ \ge \frac{t}{n}\right) &= 1 - Q_n(n+1) \end{aligned}$$

2.5. Steck recursion formula

Steck (1969) derived the recursion formula (referred to henceforth as Steck) shown below that was later listed in Shorack and Wellner (1986).

$$\begin{aligned} b_j &= \min\left(\frac{j-1+t}{n}, 1\right) \qquad \text{for } j = 1, 2, \cdots, n \\ P_0 &= 1 \\ P_1 &= b_1 \\ P_i &= b_i^i - \sum_{m=0}^{i-2} \binom{i}{m} [b_i - b_{m+1}]^{i-m} P_m \qquad \text{for } i = 2, 3, \cdots, n \\ P\left(D_n^+ \ge \frac{t}{n}\right) &= 1 - P_n \end{aligned}$$

2.6. Conover recursion formula

Conover (1972) derived a recursion formula (referred to henceforth as Conover) that simplifies to the following for a hypothesized continuous cumulative distribution $F(x)$.

$$\begin{aligned} e_0 &= 1 \\ e_k &= 1 - \sum_{j=0}^{k-1} \binom{k}{j} \left(1 - \frac{j}{n} - \frac{t}{n}\right)^{k-j} e_j \qquad \text{for } k = 1, 2, \cdots, \lfloor n - t \rfloor \\ P\left(D_n^+ \ge \frac{t}{n}\right) &= \sum_{j=0}^{\lfloor n-t \rfloor} \binom{n}{j} \left(1 - \frac{j}{n} - \frac{t}{n}\right)^{n-j} e_j \end{aligned}$$

Although not stated explicitly, this appears to be the recursion formula used by the **IMSL** and R statistical software packages.

2.7. Bolshev recursion formula

Kotelnikov and Chmaladze (1983) used the recursion formula (referred to henceforth as Bolshev) shown below that was later called the Bolshev recursion in Shorack and Wellner (1986).

$$\begin{aligned} b_j &= \min\left(\frac{j-1+t}{n}, 1\right) && \text{for } j = 1, 2, \cdots, n \\ P_0 &= 1 \\ P_i &= 1 - \sum_{m=1}^{i} \binom{i}{m} [1 - b_{i-m+1}]^m P_{i-m} && \text{for } i = 1, 2, \cdots, n \\ P\left(D_n^+ \geq \frac{t}{n}\right) &= 1 - P_n \end{aligned}$$

3. Computational and research issues

The thirteen formulae presented in the last section are complex. Consequently, implementing them raises certain computational issues that need to be studied. The following are the three major computational questions that need to be resolved before the thirteen formulae are implemented.

1. What type of computational arithmetic should be used?
2. What arithmetic form should be used to input the test statistic d^+ to the thirteen formulae?
3. What is the most efficient way to calculate the binomial coefficients?

These questions will be considered and answered in the order shown above. After these implementation questions have been resolved, the following four research questions will be answered.

1. What is the best way to calculate the bandwidth (the inverse of the cumulative sampling distribution)? The answer to this question will be used to compute and present detailed bandwidth tables.
2. What is the relationship between computation time and sample size n?
3. What is the fastest formulae?
4. What is the relationship between the accuracy of the test statistic d^+ and the computation time?

3.1. Computational options

Using current computational software, the formulae can be implemented using either rational arithmetic, arbitrary precision arithmetic, or machine precision arithmetic. Rational arithmetic stores every number as a ratio of two integers (a rational number) where each integer

can have as many decimal digits as needed to express the number exactly. Although the speed of rational arithmetic declines as the number of digits in the numerator/denominator integers increase, it has the advantage of no error as long as no irrational numbers are used. Conversely, machine precision arithmetic specifies the number of decimal digits (usually less than twenty and determined by the computer hardware) to use in computations so it is subject to roundoff error and catastrophic cancelation. Catastrophic cancelation occurs when one number is subtracted from another number of about the same value. For example, if 123.345689 is subtracted from 123.3456799 both with nine decimal digits of precision, then the result is 0.000010 with two decimal digits of precision. Although machine precision is fast, it is possible to significantly degrade the accuracy and even worse, not be aware that the accuracy has been reduced. Arbitrary precision arithmetic is like machine precision except that the number of decimal digits of precision is not dependent on the computer hardware and the user can specify the number of decimal digits of precision. Although arbitrary precision is slower than machine precision, it is faster than rational arithmetic. In addition, the software system Mathematica, Wolfram (2003), keeps track of the resulting precision rp so that for the example above, Mathematica would also know that the result 0.000010 had a precision of $rp = 2$. The trick in using arbitrary precision arithmetic is specifying the precision to be used in internal calculations (internal precision ip) so that the final answer has a specified desired precision dp or greater. In other words, the user must specify both ip and dp so that the final answer has $rp \geq dp$. Since all the K-S formulae can be modified so that no irrational numbers are used, this paper will use rational arithmetic implementations as they produce exact p values with no error. Future research will develop machine precision and arbitrary precision implementations whose accuracy will then be verified by the rational arithmetic implementations in this paper.

In terms of accuracy, rational arithmetic gives the exact probability (no error) as long as the test statistic d can be expressed exactly as a rational number; it cannot be an irrational number like $\pi/100$. The only way d^+ can be an irrational number is if the hypothesized distribution $F(x)$ for some x is an irrational number because by definition the empirical distribution $F_n(x)$ is a rational number (i/n for $i = 0, 1, 2, \ldots, n$). In such cases d can be approximated arbitrarily closely by rational numbers above and below d^+. These are then used to calculate the p value $P[D_n^+ \geq d^+]$ to any desired accuracy. Thus, rational arithmetic either provides the exact p value if d^+ is a rational number or can get as close as the user desires if d^+ is an irrational number.

3.2. Test statistic complexity

Using rational arithmetic, the sample size n and the test statistic d^+ can produce a p value with many digits in the numerator and denominator integers. For example, when $n = 200$ and $d^+ = 13/200$; d^+ has two digits in the numerator and three digits in the denominator ([2/3] numerator/denominator digits) while the corresponding p value $P\left[D_n^+ \geq d^+\right]$ has [456/459] numerator/denominator digits. To show how the number of numerator/denominator digits can grow, consider another example with $n = 2,000$ and $d^+ = 83/2000$ where d^+ has [2/4] numerator/denominator digits while the corresponding p value $P\left[D_n^+ \geq d^+\right]$ has [6599/6602] numerator/denominator digits. The large number of numerator/denominator digits for the p values $P\left[D_n^+ \geq d^+\right]$ suggest that computational time might vary with the number of numerator/denominator digits in test statistic d^+. This hypothesis is tested in Section 8.

A rational number implementation of each of the thirteen formula has two inputs, the sample size n that is by definition a rational number (an integer) and the test statistic d^+ that is a number between zero and one, $0 \leq d^+ \leq 1$. If d^+ is neither zero, one, or an irrational number, then d^+ can be expressed as either a rational number or an arbitrary precision number. For example, with $n = 100$ and $d^+ = 0.183683$, d^+ can be used in the program as the rational number $d^+ = 183683/1000000$ or as the arbitrary precision number $d^+ = 0.183683$ while all the rest of the computations in the Mathematica program are in rational arithmetic. Since Mathematica treats rational numbers and arbitrary precision numbers differently, the same Mathematica program that implements the SmirnovD formula in rational arithmetic will yield different probabilities depending on whether d^+ is used as rational number or an arbitrary precision number in the program. When used as a rational number in the Mathematica program, $d^+ = 183683/1000000$ produces a rational number probability with 597 digits in the numerator and 600 digits in the denominator ([597/600] numerator/denominator digits) that when converted to an arbitrary precision number with 20 decimal digits of accuracy yields $P\left[D_{100}^+ \geq d^+\right] = 0.0010000109813850096033$. However, when used as an arbitrary precision number in the Mathematica program, $d^+ = 0.183683$ yields a different p value, $P\left[D_{100}^+ \geq d^+\right] = 29398345/29398022169 = 0.0010000109813850110101$, than that produced by the rational number d^+. Since the correct probability is the one produced by the rational number input, the input d^+ is always converted to a rational number before it is used in any Mathematica program.

3.3. Calculating a series of binomial coefficients

Since all thirteen formulae use the binomial coefficient in almost every term, an important consideration is how to calculate them. Although the notation varies in each formulae, let $\binom{k}{j}$ represent the binomial coefficient. The four iterative formulae include the binomial coefficient in the iterative formulae for x_j and y_j so that the binomial coefficient is automatically included and need not be calculated separately. However, every one of the direct formulae and the recursion formulae use the binomial coefficient in each term of the summations. In each of these summations, the binomial coefficient $\binom{k}{j}$ for each succeeding term changes by adding one to j so that each formula uses a series of binomial coefficients. The efficiency of the method for calculating the series of binomial coefficients as rational numbers will effect the speed of the implementation for each formulae. The two following competing methods can be used to compute every binomial coefficient in the series as a rational number.

Rational Number Binomial Function (RNBF) – Use the Mathematica Binomial function to generate each coefficient as a rational number.

Rational Number Iterative Calculation (RNIC) – Use rational arithmetic to iteratively compute $\binom{k}{j}$ from $\binom{k}{j-1}$ by multiplying it by $\frac{k-j+1}{j}$.

Brown and Harvey (2006) compared the RNBF method versus the RNIC method and for a small number of terms nt found little difference in the computation times. However, for a large number of terms, the RNIC method is faster than the RNBF method and the difference

Formula Name	Type Formula	Mathematica Function Name	Listed In Section
SmirnovD	Direct	SmirnovDKS1SidedRTRational	1
DwassD	Direct	DwassDKS1SidedRTRational	2
SmirnovAltD	Direct	SmirnovAltDKS1SidedRTRational	3
DwassAltD	Direct	DwassAltDKS1SidedRTRational	4
SmirnovI	Iterative	SmirnovIKS1SidedRTRational	5
DwassI	Iterative	DwassIKS1SidedRTRational	6
SmirnovAltI	Iterative	SmirnovAltIKS1SidedRTRational	7
DwassAltI	Iterative	DwassAltIKS1SidedRTRational	8
Daniels	Recursion	DanielsKS1SidedRTProbRational	9
Noe	Recursion	NoeKS1SidedRTProbRational	10
Steck	Recursion	SteckKS1SidedRTProbRational	11
Conover	Recursion	ConoverKS1SidedRTProbRational	12
Bolshev	Recursion	BolshevKS1SidedRTProbRational	13

Table 5: Mathematica function name for the thirteen formulae listed in file `KS1SidedOneSampleRational.nb`

in time increases as the sample size n increases. This implies that as the number of terms grow, a RNBF implementation will eventually exceed the time needed by the corresponding RNIC implementation. In addition, RNBF uses the Mathematica Binomial function so any code using the RNBF method must be implemented in Mathematica while the RNIC method can be implemented using any rational arithmetic software. Thus, RNIC is more portable than RNBF and is another reason for adopting RNIC. As a result, this paper will use the RNIC method exclusively to calculate the binomial coefficients for the thirteen formulae.

4. Implementations of the thirteen formulae

All the Mathematica code generated for this paper is contained in one Mathematica file named `KS1SidedOneSampleRational.nb` which is divided into 23 sections. Each section contains one program and sample output. Table 5 contains a list of all the formulae, their type, the Mathematica function name implementing them, and the section number in file `KS1SidedOneSampleRational.nb` that contains the Mathematica code and sample output.

5. Calculating the one-sided bandwidth $d^+(n, \alpha, \rho)$

In addition to calculating the p value for hypothesis testing, the one-sided one-sample K-S cumulative sampling distribution can be used to construct a one-sided confidence band around the empirical distribution $F_n(x)$. The bandwidth of a one-sided confidence band with confidence coefficient $1-\alpha$ and sample size n is the value of the test statistic d^+ that satisfies

$P(D_n^+ \geq d^+) = \alpha$. Determining a bandwidth d^+ for a particular sample size n and confidence coefficient $1-\alpha$ means evaluating the inverse of the cumulative sampling distribution which can only be done by search techniques such as binary search. Unlike the p value, a bandwidth d^+ cannot in practice be determined exactly because the search technique may not converge to the exact value. For example, binary search with starting values of 0 and 1 would never find $d^+ = 1/3$ and would iterate forever. Thus, search techniques are designed to stop when a specified accuracy is reached. Let $d^+(n, \alpha, \rho)$ represent the bandwidth rounded to ρ significant digits for sample size n and confidence coefficient $1-\alpha$. Note that bandwidth $d^+(n, \alpha, \rho)$ is also the hypothesis testing critical value for an α level of significance.

Finding the bandwidth $d^+(n, \alpha, \rho)$ is a three step process: (1) use an approximation to find an initial value close to $d^+(n, \alpha, \rho)$, (2) use the initial value to find upper and lower bounds on $d^+(n, \alpha, \rho)$, and (3) use a search procedure to determine $d^+(n, \alpha, \rho)$ between the lower and upper bounds.

The first step will use the approximation of Maag and Dicaire (1971) to find the initial value by solving $\alpha \simeq \exp\left(\frac{-[6nd^+ + 1]^2}{18n}\right)$ for d^+ yielding $d^+ \simeq \sqrt{\frac{\ln(\alpha)}{-2n}} - \frac{1}{6n}$. Since the initial value found by the approximation in the first step should be fairly close to the actual value, the second step gradually increases the distance away from the initial value until a lower and upper bound on the actual value is found. Although there are many search techniques that can be used in the third step to determine the bandwidth, this paper will consider the two most common techniques: binary search and linear search. Note that binary search takes the midpoint between the upper and lower bounds as the next value to test while linear search uses linear interpolation to find the next value. Preliminary computational experience showed that linear search was always faster than binary search so the following linear search algorithm is used to find the bandwidth $d^+(n, \alpha, \rho)$.

Linear search algorithm for calculating the bandwidth $d^+(n, \alpha, \rho)$

Step 1 (Find Initial Value): Calculate $d^+ = \sqrt{\frac{\ln(\alpha)}{-2n}} - \frac{1}{6n}$ to ρ digits of precision. If $d^+ > 1$, set $d^+ = 1$. If $d^+ < 0$, set $d^+ = 0$. Go to Step 2.

Step 2 (Determine If Initial Value Is Lower Or Upper Bound): Calculate $p = P[D^+ \geq d^+]$. If $p > \alpha$, then d^+ is a lower bound, set $d_L^+ = d^+$ set $p_L = p$, and go to Step 3. Otherwise, d^+ is an upper bound, set $d_U^+ = d^+$, and go to Step 6.

Step 3 (Determine an Upper Bound): Convert d_L^+ to a numerator integer $dnumerator_L^+$ and a denominator integer $ddenominator_L^+$ where $ddenominator_L^+$ is a power of ten and $d_L^+ = dnumerator_L^+ / ddenominator_L^+$. If the number of digits of precision does not exceed four, $\rho \leq 4$, set increment $inc = 1$. Otherwise $\rho > 4$ and set the increment $inc = 10^{\rho-5}$. Go to Step 4.

Step 4 (Construct and Test a Possible Upper Bound): Set $dtry = dnumerator_L^+ + inc$, calculate $p = P[D^+ \geq dtry/ddenominator_L^+]$. If $p > \alpha$, then a new lower bound has been found and go to Step 5. Otherwise, the initial upper bound has been found, set $dnumerator_U^+ = dtry$, set $dnumerator_U^+ = dnumerator_L^+$, set $p_U = p$, and go to Step 9.

Step 5 (New Lower Bound Found): Set $dnumerator_L^+ = dtry$, set $p_L = p$, set $inc = inc \times 10$, and go to Step 4.

Step 6 (Determine a Lower Bound): Convert d_U^+ to a numerator integer $dnumerator_U^+$ and a denominator integer $ddenominator_U^+$ where $ddenominator_U^+$ is a power of ten and $d_U^+ = dnumerator_U^+/ddenominator_U^+$. If the number of digits of precision does not exceed four, $\rho \leq 4$, set increment $inc = 1$. Otherwise $\rho > 4$ and set the increment $inc = 10^{\rho-5}$. Go to Step 7.

Step 7 (Construct and Test a Possible Lower Bound): Set $dtry = dnumerator_U^+ - inc$, calculate $p = P[D^+ \geq dtry/ddenominator_U^+]$. If $p < \alpha$, then a new upper bound has been found and go to Step 8. Otherwise, the initial lower bound has been found, set $dnumerator_L^+ = dtry$, set $dnumerator_L^+ = dnumerator_U^+$, set $p_L = p$, and go to Step 9.

Step 8 (New Upper Bound Found): Set $dnumerator_U^+ = dtry$, set $p_U = p$, set $inc = inc \times 10$, and go to Step 7.

Step 9 (Linear Search Iteration): If $dnumerator_U^+ - dnumerator_L^+ \leq 1$, go to Step 12. Set $dtry = \lfloor dnumerator_L^+ + (dnumerator_U^+ - dnumerator_L^+) \times (p_L - \alpha)/(p_L - p_U) \rfloor$. If $dtry \geq dnumerator_U^+$, set $dtry = dnumerator_U^+ - 1$. If $dtry \leq dnumerator_L^+$, set $dtry = dnumerator_L^+ + 1$. Calculate $p = P[D^+ \geq dtry/ddenominator_L^+]$. If $p > \alpha$, then a new lower bound has been found and go to Step 10. Otherwise, a new upper bound has been found and go to Step 11.

Step 10 (Linear Search New Lower Bound): Set $dnumerator_L^+ = dtry$, $p_L = p$, and go to Step 9.

Step 11 (Linear Search New Upper Bound): Set $dnumerator_U^+ = dtry$, $p_U = p$, and go to Step 9.

Step 12 (Determine Whether to Use Lower or Upper Bound): Calculate $p = P[D^+ \geq (dnumerator_L^+ \times 10 + 5)/(ddenominator_L^+ \times 10)]$. If $p < \alpha$, then use the lower bound by setting $d^+ = dnumerator_L^+/ddenominator_L^+$ and go to Step 13. Otherwise, use the upper bound by setting $d^+ = dnumerator_U^+/ddenominator_U^+$ and go to Step 13.

Step 13 (Bandwidth Found): Terminate the algorithm with the bandwidth d^+.

The linear search algorithm is implemented using the direct formulae: SmirnovD, SmirnovAltD, DwassD, and DwassAltD. The section number in file `KS1SidedOneSampleRational.nb` containing the Mathematica function that implements the linear search algorithm for each direct formula is listed in Table 6.

Tables 7 and 8 contain computational experience of the linear search algorithms for each direct formula to find the bandwidths. Since the DwassAltD linear search algorithm is faster than the other three direct formula implementations, it will be used in the remainder of the paper to find bandwidths.

Direct Formula	In Mathematica File KS1SidedOneSampleRational.nb Mathematica Function Name	Section Number
DwassD	KS1SidedBandwidthByLinearSearchDwassD	15
DwassAltD	KS1SidedBandwidthByLinearSearchDwassAltD	14
SmirnovD	KS1SidedBandwidthByLinearSearchSmirnovD	17
SmirnovAltD	KS1SidedBandwidthByLinearSearchSmirnovAltD	16

Table 6: Direct formula implementations of the linear search algorithm to find bandwidths

The Mathematica function `KS1SidedOneSampleBandwidthsToFile` contained in Section 18 of the `KS1SidedOneSampleRational.nb` file finds bandwidths using linear search with DwassAltD and writes these bandwidths to a comma delimited file for input into Excel and a text file that can be used as the input into timing programs. The text file contains bandwidths where every digit in a bandwidth is output separately so the bandwidth can be reconstructed to any desired accuracy. These text files will be used as input files to produce the computational experience in Sections 6, 8, and 9.

Tables 9, 10, and 11 contain the bandwidths to six digits of precision ($\rho = 6$) for $\alpha = 0.2, 0.1, 0.05, 0.02, 0.01, 0.001$ and representative sample sizes from $n = 2$ through $n = 2,000$.

6. Computational experience comparing all thirteen formulae

This section compares the computer time needed to calculate the same p value by all thirteen formulae with the objective of determining the fastest formula. Using the program in Section 5, bandwidths are generated for sample sizes $n = 1000, 2500, 5000$ and confidence coefficients $\alpha = 0.001, 0.01, 0.1, 0.25, 0.5, 0.9$. The resulting bandwidths with $\rho = 20$ digits of precision are put into data file `KS1SidedOneSampleBandwidthsN1000to5000.dat` and Excel file `KS1SidedOneSampleBandwidthsN1000to5000.csv`. The six confidence coefficients for $\alpha = 0.001, 0.01, 0.1, 0.25, 0.5, 0.9$ were chosen so that the entire range of α's from 0 to 1 had some representation but the range of greatest p value interest from 0.001 to 0.1 has the most representation.

To illustrate the results, Table 12 contains the $\rho = 20$ bandwidths for both $\alpha = 0.001$ and $\alpha = 0.9$. In addition, Table 12 contains all the $\rho = 3$ bandwidths in both decimal and rational form.

6.1. Direct formulae computational experience

In comparing the computational times across various sample sizes, the question is what values of the test statistic d^+ should be used for comparison. The two alternatives are comparing the computational times for a fixed value of the test statistic d^+ or comparing the computational times needed to produce a specified p value α. The difficulty with comparing the computation times for a fixed value of the test statistic d^+ is that the p value will vary with n. For example, $P[D^+_{1,000} \geq 293/5000] \simeq 0.001$ while $P[D^+_{5,000} \geq 293/5000] \simeq 1.14493 \times 10^{-15}$. A more useful

Sample Size n	p value α	Direct Formula	Time In Seconds (Time) and Number Iterations (Iters) $\rho = 3$ Time	Iters	$\rho = 6$ Time	Iters	$\rho = 9$ Time	Iters	$\rho = 12$ Time	Iters
100	0.001	SmironvD	0.063	4	0.14	7	0.281	9	0.454	11
		SmironvAltD	0.047	4	0.141	7	0.25	9	0.39	11
	0.01	SmironvD	0.046	4	0.157	7	0.25	8	0.468	10
		SmironvAltD	0.032	4	0.125	7	0.218	8	0.407	10
	0.1	SmironvD	0.063	4	0.141	6	0.203	6	0.313	7
		SmironvAltD	0.047	4	0.093	6	0.172	6	0.281	7
	0.25	SmironvD	0.078	5	0.156	6	0.266	7	0.406	8
		SmironvAltD	0.047	5	0.125	6	0.203	7	0.36	8
	0.5	SmironvD	0.063	4	0.156	6	0.265	7	0.407	8
		SmironvAltD	0.062	4	0.125	6	0.219	7	0.375	8
	0.9	SmironvD	0.093	5	0.157	6	0.281	7	0.453	8
		SmironvAltD	0.078	5	0.141	6	0.234	7	0.375	8
250	0.001	SmironvD	0.344	4	0.906	6	2.406	9	4.578	11
		SmironvAltD	0.156	4	0.61	6	1.719	9	3.562	11
	0.01	SmironvD	0.407	4	1.078	6	2.422	8	4.047	9
		SmironvAltD	0.219	4	0.719	6	1.796	8	3.188	9
	0.1	SmironvD	0.421	4	0.954	5	1.812	6	2.734	6
		SmironvAltD	0.219	4	0.641	5	1.312	6	2.157	6
	0.25	SmironvD	0.516	5	1.094	6	2.265	7	3.204	7
		SmironvAltD	0.25	5	0.703	6	1.687	7	2.5	7
	0.5	SmironvD	0.406	4	1.172	6	2.25	7	3.359	7
		SmironvAltD	0.219	4	0.797	6	1.656	7	2.656	7
	0.9	SmironvD	0.469	4	1.219	6	2.39	7	4.328	9
		SmironvAltD	0.266	4	0.812	6	1.781	7	3.438	9
500	0.001	SmironvD	2.016	4	6.703	7	14.734	8	26.297	9
		SmironvAltD	0.906	4	4.062	7	10.469	8	19.406	9
	0.01	SmironvD	2.047	4	6.485	6	15.406	8	27.219	9
		SmironvAltD	0.797	4	3.797	6	10.672	8	20.093	9
	0.1	SmironvD	2.078	4	6.375	6	10.078	5	18.282	6
		SmironvAltD	0.907	4	3.843	6	6.938	5	13.453	6
	0.25	SmironvD	2.609	5	5.906	5	13.203	6	17.922	6
		SmironvAltD	1.062	5	3.563	5	8.625	6	12.641	6
	0.5	SmironvD	2.719	5	5.719	5	10.015	5	17.938	6
		SmironvAltD	1.078	5	3.5	5	6.828	5	13.031	6
	0.9	SmironvD	2.891	4	8.656	6	16	7	27.281	8
		SmironvAltD	1.531	4	5.61	6	11.656	7	21.141	8

Note: All timings on a Pentium IV running at 2.4 GHz.

Table 7: SmirnovD and SmirnovAltD linear search algorithms calculating bandwidth $d^+(n, \alpha, \rho)$

Sample Size n	p value α	Direct Formula	Time In Seconds (Time) and Number Iterations (Iters) $\rho = 3$ Time	Iters	$\rho = 6$ Time	Iters	$\rho = 9$ Time	Iters	$\rho = 12$ Time	Iters
1,000	0.001	DwassD	0.469	4	1.75	6	4.765	8	8.407	9
		DwassAltD	0.172	4	0.922	6	3.172	8	6.093	9
	0.01	DwassD	0.422	4	1.25	5	2.765	6	5.406	7
		DwassAltD	0.157	4	0.734	5	1.813	6	3.921	7
	0.1	DwassD	0.329	4	0.906	5	1.656	5	3.172	6
		DwassAltD	0.141	4	0.531	5	1.11	5	2.281	6
	0.25	DwassD	0.281	4	0.64	5	1.625	6	2.516	6
		DwassAltD	0.125	4	0.375	5	1.093	6	1.828	6
	0.5	DwassD	0.203	4	0.625	6	0.969	5	1.859	6
		DwassAltD	0.079	4	0.39	6	0.672	5	1.375	6
	0.9	DwassD	0.094	4	0.281	6	0.641	7	1.109	8
		DwassAlt	0.047	4	0.203	6	0.469	7	0.906	8
2,500	0.001	DwassD	5.5	4	13.438	5	29	6	60.047	7
		DwassAltD	1.734	4	5.735	5	14.906	6	36.75	7
	0.01	DwassD	3.609	4	11.641	5	25.047	6	46.25	7
		DwassAltD	0.157	4	0.734	5	1.813	6	3.921	7
	0.1	DwassD	2.968	4	6.813	4	17.187	6	26.703	6
		DwassAltD	0.875	4	3	4	9.109	6	15.969	6
	0.25	DwassD	2.282	4	5.359	5	11.063	5	21.953	6
		DwassAltD	0.687	4	2.172	5	5.891	5	13.406	6
	0.5	DwassD	1.609	4	4.328	5	7.875	5	15.672	6
		DwassAltD	0.516	4	1.844	5	4.265	5	9.703	6
	0.9	DwassD	0.828	4	2.094	5	4.766	6	7.171	6
		DwassAltD	0.328	4	1.094	5	2.891	6	4.765	6
5,000	0.001	DwassD	27.829	4	74.593	5	158.86	6	307.14	7
		DwassAltD	7.188	4	27.812	5	76.157	6	174.437	7
	0.01	DwassD	16.391	4	68.344	6	136.437	6	240.375	7
		DwassAltD	2.625	4	25.25	6	66.422	6	135.438	7
	0.1	DwassD	12.828	4	34.125	4	77.953	5	151.047	6
		DwassAltD	2.671	4	12.766	4	37.969	5	86.437	6
	0.25	DwassD	11.828	4	31.61	5	61.281	5	115.469	6
		DwassAltD	3.047	4	11.797	5	29.984	5	66.25	6
	0.5	DwassD	10.687	4	30.594	6	61.375	6	91.937	6
		DwassAltD	3.141	4	12.5	6	32.281	6	54.532	6
	0.9	DwassD	4	4	11.844	5	24.781	6	36.625	6
		DwassAltD	1.343	4	5.422	5	13.797	6	22.594	6

Note: All timings on a Pentium IV running at 2.4 GHz.

Table 8: DwassD and DwassAltD linear search algorithms calculating bandwidth $d^+(n, \alpha, \rho)$

Sample	Bandwidth $d^+(n, \alpha, \rho = 6)$					
Size n	$\alpha = 0.2$	$\alpha = 0.1$	$\alpha = 0.05$	$\alpha = 0.02$	$\alpha = 0.01$	$\alpha = 0.001$
2	.552786	.683772	.776393	.858579	.900000	.968377
3	.472674	.564810	.636045	.728558	.784557	.900000
4	.412407	.492653	.565216	.640745	.688870	.822172
5	.370169	.446980	.509449	.579665	.627180	.750000
6	.340585	.410373	.467993	.534303	.577407	.695706
7	.317415	.381476	.436069	.497468	.538440	.650714
8	.298083	.358313	.409623	.467651	.506543	.613676
9	.281849	.339102	.387464	.442728	.479596	.582099
10	.268074	.322602	.368663	.421350	.456624	.555002
11	.256238	.308292	.352421	.402834	.436703	.531346
12	.245918	.295770	.338151	.386604	.419178	.510472
13	.236761	.284698	.325490	.372204	.403621	.491890
14	.228543	.274807	.314170	.359308	.389695	.475202
15	.221128	.265886	.303973	.347677	.377127	.460107
16	.214400	.257784	.294720	.337119	.365709	.446371
17	.208264	.250387	.286269	.327476	.355275	.433799
18	.202638	.243601	.278511	.318621	.345693	.422236
19	.197453	.237346	.271357	.310453	.336852	.411555
20	.192652	.231555	.264734	.302887	.328661	.401649
21	.188188	.226173	.258577	.295853	.321044	.392427
22	.184023	.221153	.252835	.289292	.313936	.383816
23	.180125	.216455	.247462	.283151	.307283	.375750
24	.176468	.212048	.242420	.277388	.301039	.368174
25	.173028	.207902	.237677	.271966	.295163	.361040
26	.169784	.203992	.233205	.266852	.289621	.354308
27	.166718	.200297	.228977	.262018	.284381	.347940
28	.163814	.196798	.224974	.257440	.279417	.341905
29	.161059	.193478	.221175	.253094	.274706	.336174
30	.158440	.190321	.217563	.248964	.270227	.330724
31	.155945	.187316	.214125	.245030	.265962	.325531
32	.153566	.184450	.210845	.241278	.261893	.320577
33	.151294	.181712	.207713	.237695	.258007	.315843
34	.149121	.179094	.204718	.234268	.254290	.311314
35	.147039	.176587	.201849	.230985	.250730	.306975
36	.145044	.174183	.199098	.227838	.247316	.302813
37	.143128	.171876	.196458	.224817	.244038	.298817
38	.141287	.169659	.193921	.221913	.240889	.294975
39	.139516	.167526	.191480	.219120	.237858	.291279
40	.137810	.165472	.189130	.216431	.234940	.287718
41	.136167	.163492	.186865	.213838	.232128	.284286
42	.134581	.161582	.184680	.211338	.229414	.280974
43	.133049	.159739	.182570	.208923	.226794	.277775
44	.131570	.157957	.180532	.206590	.224263	.274684
45	.130139	.156234	.178560	.204333	.221814	.271694
46	.128754	.154567	.176653	.202150	.219445	.268800
47	.127413	.152952	.174805	.200035	.217150	.265997
48	.126113	.151388	.173015	.197986	.214926	.263280
49	.124853	.149870	.171279	.195999	.212769	.260646

Table 9: Bandwidth $d^+(n, \alpha, \rho = 6)$ to six digits of precision for $n = 2$ to $n = 49$

Sample	Bandwidth $d^+(n, \alpha, \rho = 6)$					
Size n	$\alpha = 0.2$	$\alpha = 0.1$	$\alpha = 0.05$	$\alpha = 0.02$	$\alpha = 0.01$	$\alpha = 0.001$
50	.123630	.148398	.169594	.194070	.210677	.258089
60	.113108	.135735	.155106	.177484	.192675	.236081
70	.104898	.125858	.143806	.164548	.178632	.218900
80	.0982602	.117874	.134673	.154091	.167280	.205005
90	.0927478	.111245	.127091	.145411	.157855	.193466
100	.0880746	.105627	.120666	.138054	.149868	.183683
120	.0805279	.0965573	.110293	.126179	.136974	.167887
140	.0746462	.0894905	.102212	.116928	.126930	.155578
160	.0698946	.0837829	.0956867	.109458	.118818	.145637
180	.0659516	.0790476	.0902733	.103261	.112090	.137390
200	.0626109	.0750364	.0856880	.0980124	.106391	.130404
220	.0597330	.0715815	.0817390	.0934923	.101483	.124388
240	.0572200	.0685651	.0782913	.0895463	.0971989	.119135
260	.0550007	.0659015	.0752471	.0860622	.0934160	.114497
280	.0530219	.0635268	.0725333	.0829563	.0900438	.110363
300	.0512430	.0613923	.0700941	.0801648	.0870129	.106647
320	.0496325	.0594599	.0678861	.0776379	.0842694	.103283
340	.0481654	.0576997	.0658748	.0753362	.0817705	.100220
360	.0468214	.0560875	.0640327	.0732283	.0794819	.0974138
380	.0455844	.0546036	.0623373	.0712882	.0773755	.0948313
400	.0444408	.0532319	.0607700	.0694949	.0754286	.0924442
420	.0433794	.0519588	.0593156	.0678307	.0736217	.0902290
440	.0423907	.0507732	.0579611	.0662807	.0719391	.0881660
460	.0414669	.0496653	.0566954	.0648326	.0703669	.0862385
480	.0406012	.0486271	.0555094	.0634756	.0688936	.0844322
500	.0397876	.0476515	.0543950	.0622005	.0675094	.0827351
520	.0390212	.0467325	.0533452	.0609994	.0662054	.0811365
540	.0382975	.0458648	.0523540	.0598655	.0649743	.0796272
560	.0376128	.0450438	.0514163	.0587926	.0638096	.0781992
580	.0369636	.0442655	.0505272	.0577754	.0627054	.0768454
600	.0363470	.0435262	.0496829	.0568094	.0616567	.0755597
620	.0357603	.0428229	.0488795	.0558904	.0606590	.0743366
640	.0352012	.0421527	.0481140	.0550146	.0597082	.0731710
660	.0346676	.0415130	.0473834	.0541788	.0588009	.0720586
680	.0341576	.0409017	.0466852	.0533800	.0579338	.0709956
700	.0336695	.0403166	.0460170	.0526156	.0571040	.0699783
720	.0332018	.0397560	.0453767	.0518832	.0563088	.0690035
740	.0327531	.0392182	.0447625	.0511806	.0555461	.0680684
760	.0323222	.0387017	.0441726	.0505058	.0548136	.0671704
780	.0319079	.0382051	.0436055	.0498570	.0541093	.0663071
800	.0315091	.0377271	.0430597	.0492327	.0534316	.0654762
820	.0311250	.0372667	.0425339	.0486312	.0527787	.0646758
840	.0307546	.0368228	.0420269	.0480513	.0521492	.0639041
860	.0303971	.0363944	.0415377	.0474917	.0515417	.0631595
880	.0300519	.0359807	.0410652	.0469513	.0509550	.0624403
900	.0297182	.0355807	.0406085	.0464289	.0503880	.0617451

Table 10: Bandwidth $d^+(n, \alpha, \rho = 6)$ to six digits of precision for $n = 50$ to $n = 900$

Sample	Bandwidth $d^+(n, \alpha, \rho = 6)$					
Size n	$\alpha = 0.2$	$\alpha = 0.1$	$\alpha = 0.05$	$\alpha = 0.02$	$\alpha = 0.01$	$\alpha = 0.001$
920	.0293953	.0351939	.0401667	.0459235	.0498394	.0610727
940	.0290828	.0348194	.0397391	.0454344	.0493084	.0604217
960	.0287801	.0344566	.0393249	.0449606	.0487941	.0597912
980	.0284866	.0341050	.0389233	.0445013	.0482955	.0591801
1000	.0282020	.0337639	.0385338	.0440558	.0478120	.0585873
1050	.0275262	.0329541	.0376092	.0429982	.0466639	.0571800
1100	.0268969	.0322000	.0367481	.0420133	.0455949	.0558695
1150	.0263089	.0314955	.0359437	.0410933	.0445962	.0546453
1200	.0257579	.0308353	.0351899	.0402312	.0436604	.0534983
1250	.0252402	.0302151	.0344818	.0394212	.0427812	.0524206
1300	.0247525	.0296309	.0338147	.0386583	.0419532	.0514056
1350	.0242922	.0290793	.0331850	.0379381	.0411715	.0504474
1400	.0238566	.0285575	.0325893	.0372568	.0404319	.0495409
1450	.0234437	.0280629	.0320245	.0366109	.0397308	.0486816
1500	.0230515	.0275931	.0314882	.0359976	.0390651	.0478656
1550	.0226784	.0271462	.0309780	.0354140	.0384317	.0470893
1600	.0223229	.0267203	.0304918	.0348580	.0378282	.0463496
1650	.0219836	.0263139	.0300278	.0343275	.0372523	.0456437
1700	.0216594	.0259256	.0295845	.0338204	.0367020	.0449692
1750	.0213491	.0255539	.0291602	.0333352	.0361753	.0443237
1800	.0210518	.0251978	.0287536	.0328703	.0356708	.0437052
1850	.0207666	.0248562	.0283637	.0324244	.0351867	.0431120
1900	.0204927	.0245281	.0279892	.0319961	.0347219	.0425423
1950	.0202294	.0242128	.0276292	.0315844	.0342751	.0419946
2000	.0199760	.0239092	.0272827	.0311882	.0338450	.0414676

Table 11: Bandwidth $d^+(n, \alpha, \rho = 6)$ to six digits of precision for $n = 920$ to $n = 2,000$

approach is to compare the computation times needed to produce the same p value across various sample sizes. In order to do this, we need to calculate the value of the test statistic that yields a specified p value α for a sample size n. In other words, we will use the bandwidth $d^+(n, \alpha, \rho)$ where $P[D_n^+ \geq d^+(n, \alpha, \rho)] \simeq \alpha$ (see Section 5).

For $\rho = 3$, the Mathematica function `TimingKS1SidedOneSampleRationalDirectFormulae` contained in Section 19 of the `KS1SidedOneSampleRational.nb` file inputs the test statistics listed Table 12 and produces the timings in Table 13. The fastest direct formula in Table 13 was DwassAltD followed in order by DwassD, SmirnovAltD, and SmirnovD. Since the number of terms in the SmirnovD and SmirnovAltD formulae for the same sample size n increase with increasing α, we would expect that the time would also increase with α. This is the pattern followed by the times in Table 13 with two exceptions which will be dealt with in Section 8: the time decreases for $n = 1,000$ going from $\alpha = 0.25$ to $\alpha = 0.5$ and the time also decreases for $n = 2,500$ going from $\alpha = 0.01$ to $\alpha = 0.1$.

6.2. Iterative formulae computational experience

For $\rho = 3$, the Mathematica function `TimingKS1SidedDirectVersusIterFormulae` contained in Section 20 of the `KS1SidedOneSampleRational.nb` file inputs the test statistics listed

Sample Size n	Bandwidth $d^+(n, \alpha, \rho = 20)$ $\alpha = 0.001$	$\alpha = 0.9$
1,000	0.058587291690890652166	0.0070941136544958142815
2,500	0.037098569056693520814	0.0045244445380207103595
5,000	0.026247865445378139343	0.0032128340598027961926

Sample Size n	Bandwidth $d^+(n, \alpha, \rho = 3)$ $\alpha = 0.001$	$\alpha = 0.01$	$\alpha = 0.1$	$\alpha = 0.25$	$\alpha = 0.5$	$\alpha = 0.9$
1,000	0.0586	0.0478	0.0338	0.0262	0.0185	0.00709
	293/5000	239/5000	169/5000	131/5000	37/2000	709/100000
2,500	0.0371	0.0303	0.0214	0.0166	0.0117	0.00452
	371/10000	303/10000	107/5000	83/5000	117/10000	113/25000
5,000	0.0262	0.0214	0.0151	0.0117	0.00829	0.00321
	131/5000	107/5000	151/10000	117/10000	829/10000	321/10000

Table 12: Bandwidth $d^+(n, \alpha, \rho)$ to produce $P\left[D_n^+ \geq d^+(n, \alpha, \rho)\right] \simeq \alpha$

Sample Size n	Formula	Time in Seconds to calculate $P\left[D_n^+ \geq d^+(n, \alpha, 3)\right]$ $\alpha = 0.001$	$\alpha = 0.01$	$\alpha = 0.1$	$\alpha = 0.25$	$\alpha = 0.5$	$\alpha = 0.9$
1,000	SmirnovD	2.609	2.641	2.656	2.703	1.891	4.188
	SmirnovAltD	0.828	0.843	0.844	0.860	0.406	1.922
	DwassD	0.125	0.093	0.063	0.047	0.047	0.031
	DwassAltD	0.047	0.047	0.015	0.016	0.000	0.016
2,500	SmirnovD	31.875	32.046	30.282	30.500	32.593	46.469
	SmirnovAltD	6.219	6.266	4.656	4.687	6.360	11.515
	DwassD	0.922	0.734	0.500	0.390	0.282	0.157
	DwassAltD	0.203	0.157	0.094	0.062	0.063	0.031
5,000	SmirnovD	203.969	204.594	225.344	225.906	364.453	371.469
	SmirnovAltD	14.656	14.734	31.422	32.063	90.782	90.797
	DwassD	4.234	3.438	2.547	1.969	2.218	0.875
	DwassAltD	0.312	0.250	0.375	0.297	0.547	0.234

Note: All timings on a Pentium IV running at 2.4 GHz.

Table 13: Time in seconds for direct formulae to calculate $P\left[D_n^+ \geq d^+(n, \alpha, 3)\right]$ using rational arithmetic

Sample Size n	p value α	Test Statistic $d^+(n,\alpha,3)$	Time in Seconds to Compute $P[D_n^+ \geq d^+(n,\alpha,3)]$ for Formula Smirnov Direct	Smirnov Iter	SmirnovAlt Direct	SmirnovAlt Iter	Dwass Direct	Dwass Iter	DwassAlt Direct	DwassAlt Iter
1,000	0.001	0.0586	2.672	4.266	0.844	1.828	0.125	0.219	0.046	0.079
	0.01	0.0478	2.640	4.313	0.843	1.844	0.110	0.187	0.031	0.063
	0.1	0.0338	2.687	4.344	0.859	1.860	0.062	0.125	0.016	0.047
	0.25	0.0262	2.719	4.359	0.859	1.875	0.063	0.109	0.016	0.031
	0.5	0.0185	1.922	3.437	0.422	1.282	0.031	0.047	0.015	0.016
	0.9	0.00709	4.250	7.406	1.953	3.875	0.032	0.046	0.016	0.016
2,500	0.001	0.0371	31.875	59.797	6.250	19.140	0.922	1.813	0.203	0.375
	0.01	0.0303	32.062	59.766	6.281	19.234	0.750	1.454	0.156	0.312
	0.1	0.0214	30.282	55.187	4.688	15.875	0.484	0.953	0.078	0.156
	0.25	0.0166	30.391	55.453	4.703	15.875	0.375	0.735	0.062	0.125
	0.5	0.0117	32.516	60.672	6.422	19.500	0.265	0.547	0.063	0.125
	0.9	0.00452	46.500	81.265	11.516	27.484	0.156	0.282	0.047	0.062
5,000	0.001	0.0262	203.609	361.234	14.532	74.937	4.235	8.109	0.312	0.688
	0.01	0.0214	204.343	362.703	14.563	74.969	3.453	6.640	0.250	0.547
	0.1	0.0151	224.938	428.328	31.047	112.187	2.578	5.094	0.375	0.735
	0.25	0.0117	225.578	429.797	31.093	112.141	1.969	3.937	0.297	0.547
	0.5	0.00829	363.484	668.344	89.422	221.594	2.250	4.359	0.547	1.031
	0.9	0.00321	364.781	668.907	89.703	221.906	0.875	1.672	0.234	0.438

Note: All timings on a Pentium IV running at 2.4 GHz.

Table 14: Comparing time in seconds for direct and iterative formulae using rational arithmetic

Table 12 and produces the timings shown in Table 14 for all direct and iterative formulae. In this timing program, the p values produced by the direct and iterative formulae for the same sample size n and test statistic d^+ are compared and an error message is written if they are not all exactly equal. For all the computational experience performed, no error message was ever generated which is an indication that the Mathematica implementations of the direct and iterative formulae are correct. The results clearly show that the direct formulae are faster than their corresponding iterative formulae. Since the DwassAltD direct formula is the fastest of all the direct and iterative formulae, it will be used as a comparison in the computational experience of the recursion formulae.

6.3. Recursion formulae computational experience

For $\rho = 3$, the Mathematica function `TimingKS1SidedRecursionFormulaeDwassAltD` contained in Section 21 of the `KS1SidedOneSampleRational.nb` file inputs the test statistics and produces the timings for the recursion formulae and DwassAltD. Preliminary testing of the recursion formulae indicated they are much slower than the direct and iterative formulae. Specifically, the computer times needed by the recursion formulae for the sample sizes $n = 1000, 2500, 5000$ used for the computational experience of the direct and iterative formulae, are excessive. Consequently, the computational experience for the recursion formulae and the direct formula DwassAltD reported in Table 15 are for the sample sizes $n = 100, 200, 300, 400, 500$. The fastest recursion formulae are Conover and Steck with Conover being faster for small α and Steck being faster for large α. However, Conover and Steck are much slower than the direct formula DwassAltD. Since each direct formula is faster than its iterative formula counterpart, and since each direct formula is faster than every recursion formula, the rest of the paper will concentrate on the direct formulae.

7. Rational number precision for the test statistic and p value

For various sample sizes n and test statistics d^+ chosen to give a good representation of the full range of the sampling distribution, Table 16 reports the approximate probability (p value) to four digits and the number of digits in the numerator and denominator integers of the rational number for the actual p value $P[D_n^+ \geq d^+]$. Table 16 also reports the time in seconds to compute the actual p value for the four direct formulae (SmirnovD, SmirnovAltD, DwassD, and DwassAltD). As the sample size n increases, the computational time as well as the number of numerator/denominator digits in the actual p value $P[D_n^+ \geq d^+]$ also increases. For a fixed sample size n, the number of numerator/denominator digits in the actual p value increases as the value of $P[D_n^+ \geq d^+]$ increases. However, for a fixed sample size n, the relationship between computational time and the p value varies across the formulae. Specifically, the computational time for SmirnovD and SmirnovAltD increases with increasing p values, while conversely the computational time for DwassD and DwassAltD decreases with increasing p values. An exception occurs for sample size $n = 3,000$ and approximate p value 0.1008. The reason for the exception is probably due to the fact that the number of numerator/denominator digits of the test statistic d^+ for approximate p value 0.1008 is less than that for the approximate p values 0.001 and 0.4996. If this is the reason for the exception, then for a fixed sample size n and a fixed confidence coefficient α, computation time should increase as the rational number precision ρ increases. This hypothesis is tested in the next section.

8. Computational times comparing rational number precisions

As stated in Section 7, the large number of numerator/denominator digits for the p values $P[D_n^+ \geq d^+]$ suggests computational time varies with the number of numerator/denominator digits in test statistic d^+. Specifically, for a fixed sample size n and a fixed confidence coefficient α, computation time should increase as the rational number precision ρ increases. Using function `TimingKS1SidedOneSampleRationalDirectFormulaeBYrnpForAlpha` contained in

Sample Size n	Formula	Time in Seconds to calculate $P\left[D_n^+ \geq d^+(n,\alpha,3)\right]$ $\alpha = 0.001$	$\alpha = 0.01$	$\alpha = 0.1$	$\alpha = 0.25$	$\alpha = 0.5$	$\alpha = 0.9$
100	Daniels	0.516	0.438	0.531	0.609	0.656	0.657
	Noe	13.578	12.828	13.406	13.985	14.453	13.922
	Steck	0.453	0.375	0.437	0.516	0.500	0.500
	Conover	0.297	0.266	0.391	0.484	0.563	0.578
	Bolshev	0.469	0.359	0.469	0.562	0.594	0.578
	DwassAltD	0.000	0.000	0.000	0.000	0.000	0.000
200	Daniels	2.687	3.500	2.735	4.609	4.500	4.453
	Noe	148.828	154.438	147.984	157.016	154.828	151.391
	Steck	2.422	3.000	2.438	3.437	3.344	3.328
	Conover	1.781	2.500	2.078	3.719	3.735	4.015
	Bolshev	2.375	3.250	2.390	4.250	4.078	4.094
	DwassAltD	0.000	0.000	0.000	0.000	0.000	0.000
300	Daniels	13.485	13.546	17.000	10.875	17.141	17.938
	Noe	724.765	716.406	729.266	696.422	711.469	694.937
	Steck	10.938	10.843	12.454	8.968	12.125	12.000
	Conover	9.172	9.765	13.391	8.453	14.563	16.172
	Bolshev	12.344	12.406	15.750	9.391	15.828	16.406
	DwassAltD	0.000	0.000	0.015	0.000	0.000	0.000
400	Daniels	37.469	37.781	37.953	38.047	29.594	38.219
	Noe	2145.380	2124.060	2078.110	2380.000	2100.190	1994.030
	Steck	28.344	28.016	27.453	27.125	22.750	26.203
	Conover	26.766	28.141	30.391	31.641	25.046	34.829
	Bolshev	35.297	35.375	35.469	35.531	26.875	35.625
	DwassAltD	0.000	0.015	0.000	0.000	0.016	0.000
500	Daniels	79.531	58.625	80.500	80.859	50.032	103.234
	Noe	5440.580	5188.610	5290.190	5268.810	5058.030	5158.560
	Steck	62.860	50.109	61.547	61.047	46.140	68.594
	Conover	57.015	43.360	64.781	66.812	43.500	94.141
	Bolshev	74.031	52.360	74.625	74.579	46.515	96.688
	DwassAltD	0.031	0.000	0.000	0.016	0.000	0.000

Note: All timings on a Pentium IV running at 2.4 GHz.

Table 15: Time in seconds for recursion formulae using rational arithmetic

Sample Size n	Test Statistic d^+	Approx Prob	Number Digits	Time in Seconds to Compute $P[D_n^+ \geq d^+]$ for Formula SmirnovD	SmirnovAltD	DwassD	DwassAltD
200	13/100	0.001	456/459	0.031	0.016	0.000	0.000
	3/40	0.1002	456/457	0.031	0.016	0.000	0.000
	51/1250	0.5003	737/737	0.078	0.047	0.000	0.000
	77/5000	0.9003	740/740	0.078	0.031	0.000	0.000
1,000	293/5000	0.001	3696/3699	3.687	1.094	0.188	0.047
	169/5000	0.0995	3698/3699	3.750	1.110	0.093	0.032
	37/2000	0.4982	3301/3301	2.578	0.531	0.031	0.016
	709/100000	0.9001	5000/5001	6.015	2.625	0.031	0.016
2,000	83/2000	0.001	6599/6602	16.000	1.640	0.531	0.063
	239/10000	0.1002	7999/8000	26.812	7.031	0.484	0.125
	131/10000	0.499	8000/8000	27.062	7.094	0.265	0.078
	101/20000	0.9	8603/8603	29.484	8.640	0.109	0.047
3,000	339/10000	0.001	13426/13429	104.422	26.032	2.782	0.672
	39/2000	0.1008	11331/11332	66.797	10.547	0.984	0.172
	107/10000	0.4996	13431/13431	106.406	26.500	0.859	0.219
	207/50000	0.8998	15527/15527	148.016	46.734	0.468	0.156

Approx Prob: 4 Digit Approximate Probability $P[D_n^+ \geq d^+]$
Number Digits: Number Digits in $P[D_n^+ \geq d^+]$ Numerator/Denominator
Note: All timings on a Pentium IV running at 2.4 GHz.

Table 16: Time in seconds for direct formulae to calculate $P[D_n^+ \geq d^+]$ using rational arithmetic

Section 22 of the `KS1SidedOneSampleRational.nb` file to test this conjecture, the computational time for all direct formulae to calculate the p value $P[D_n^+ \geq d^+(n, \alpha, \rho)]$ for sample sizes $n = 1000, 2000, 3000$, for rational number precisions $\rho = 3, 6, 9, 12, 15$, and for confidence coefficients $\alpha = 0.001, 0.9$, is recorded in Table 17. This table shows that for all direct formulae (SmirnovD, SmirnovAltD, DwassD, and DwassAltD), the hypothesis is correct, that is, computation time significantly increases as the rational number precision ρ increases.

The two exceptions noted in Section 6.1 can now be explained. In Table 13, the two exceptions are (1) the computational time decreases for $n = 1,000$ going from $\alpha = 0.25$ to $\alpha = 0.5$ and (2) the computational time also decreases for $n = 2,500$ going from $\alpha = 0.01$ to $\alpha = 0.1$. As shown above, computational time decreases as the number of numerator/denominator digits decreases in the test statistic d^+ so these exceptions may be caused by the number of

Sample Size n	α	Formula	Time in Seconds to Calculate $P\left[D_n^+ \geq d^+(n,\alpha,\rho)\right]$ $\rho = 3$	$\rho = 6$	$\rho = 9$	$\rho = 12$	$\rho = 15$
1,000	0.001	SmirnovD	2.657	7.140	13.297	21.125	30.813
		SmirnovAltD	0.843	4.156	9.000	15.657	23.938
		DwassD	0.125	0.328	0.625	1.000	1.484
		DwassAltD	0.032	0.203	0.406	0.734	1.125
2,000	0.001	SmirnovD	11.250	45.859	85.375	150.828	203.985
		SmirnovAltD	1.218	23.266	52.609	105.516	151.219
		DwassD	0.375	1.500	2.844	5.015	6.750
		DwassAltD	0.031	0.735	1.703	3.422	4.875
3,000	0.001	SmirnovD	69.375	176.828	330.547	519.938	688.656
		SmirnovAltD	18.672	84.438	194.782	345.297	482.765
		DwassD	1.828	4.735	8.938	14.188	18.719
		DwassAltD	0.484	2.110	5.000	9.031	12.735
1,000	0.9	SmirnovD	4.266	9.344	14.187	19.500	35.859
		SmirnovAltD	1.969	5.797	9.687	14.110	28.281
		DwassD	0.016	0.047	0.078	0.093	0.203
		DwassAltD	0.000	0.031	0.047	0.078	0.156
2,000	0.9	SmirnovD	20.203	64.578	113.188	181.484	257.547
		SmirnovAltD	6.375	37.016	74.906	130.750	195.406
		DwassD	0.079	0.250	0.437	0.687	0.984
		DwassAltD	0.032	0.140	0.281	0.500	0.750
3,000	0.9	SmirnovD	97.828	228.203	392.969	623.234	891.422
		SmirnovAltD	32.938	119.094	241.797	424.985	647.532
		DwassD	0.297	0.688	1.219	1.922	2.750
		DwassAltD	0.109	0.359	0.734	1.281	1.969

Note: All timings on a Pentium IV running at 2.4 GHz.

Table 17: Time in seconds for direct formulae to calculate $P\left[D_n^+ \geq d^+(n,\alpha,\rho)\right]$ for $\alpha = 0.001$ and $\alpha = 0.9$

Sample Size n	Digits of Precision ρ	Time in Seconds for DwassAltD to Calculate $P\left[D_n^+ \geq d^+(n,\alpha,\rho)\right]$ $\alpha = 0.001$	$\alpha = 0.01$	$\alpha = 0.1$	$\alpha = 0.25$	$\alpha = 0.5$	$\alpha = 0.9$
10,000	3	1.406	1.156	0.828	2.844	2.000	0.750
	6	23.657	19.203	12.922	14.453	7.875	3.797
20,000	3	6.125	5.047	14.078	11.000	2.203	3.359
	6	89.656	98.250	51.437	78.563	54.562	21.329
30,000	3	14.750	50.703	42.485	7.312	23.766	8.844
	6	284.093	462.969	326.750	253.938	98.625	70.015
40,000	3	113.453	92.640	65.829	51.406	37.062	15.891
	6	640.469	590.281	190.516	361.406	255.266	73.328
50,000	3	81.047	27.140	20.063	16.203	27.406	7.031
	6	1213.360	989.047	696.047	538.359	220.188	150.453

Note: All timings on a Pentium IV running at 2.4 GHz.

Table 18: Time in seconds for DwassAltD to calculate $P\left[D_n^+ \geq d^+(n,\alpha,\rho)\right]$ using rational arithmetic

numerator/denominator digits in the test statistic d^+ decreasing with increasing α. This is the case in the first exception because bandwidth $d^+(1000, 0.25, 3) = 0.0262 = 131/5000$ has more digits in the numerator than bandwidth $d^+(1000, 0.5, 3) = 0.0185 = 37/2000$.

9. The fastest formula, DwassAltD

The computational experience in the previous sections show that direct formula DwassAltD is the fastest. The Mathematica function `TimingKS1SidedOneSampleRationalDwassAltD` contained in Section 23 of the `KS1SidedOneSampleRational.nb` file produces timings for the direct formula DwassAltD. To get a better idea of the efficiency of DwassAltD, Table 18 contains computational experience for large sample sizes from $n = 10,000$ to $n = 50,000$ and two rational number precisions $\rho = 3$ and $\rho = 6$. As expected the computational times increase as ρ increases.

10. Conclusions

Thirteen different formulae for calculating the one-sided one-sample K-S p value were evaluated. These consisted of seven formulae (two direct formulae, SmirnovD and DwassD, and the five recursion formulae of Daniels, Noe, Steck, Conover and Bolshev) that were identified after an extensive literature search. In addition, two alternate direct formulae, SmirnovAltD

and DwassAltD, and the iterative versions of the four direct methods (SmirnovI, DwassI, SmirnovAltI, and DwassAltI) were derived. All thirteen formulae yielded identical p values but computational times varied considerably. The evaluation of these times for such a large number of formulae proceeded in three cascading stages. First, of the four direct formulae, DwassAltD is the fastest. Second, the times for DwassAltD are faster than the times for the four iterative formulae (SmirnovI, DwassI, SmirnovAltI, and DwassAltI). Finally, DwassAltD is faster than the recursion formulae of Daniels, Noe, Steck, Conover and Bolshev where Noe is the most inefficient recursion formula. For DwassAltD, the computational time to calculate the bandwidth $d^+(n, \alpha, \rho)$ increases with increasing digits of precision ρ. Since most applications will not need critical values with more than six digits of precision, the bandwidths for ($\rho = 6$), for representative sample sizes from $n = 2$ to $n = 2000$, and p values $\alpha = 0.2, 0.1, 0.05, 0.02, 0.01, 0.001$ were tabulated. To get a better sense of the efficiency of DwassAltD, the computational times to calculate p values for large sample sizes ($n = 10,000\,(10,000)\,50,000$), $\alpha = 0.001$, 0.01, 0.1, 0.25, 0.5, 0.9, and $\rho = 3$ and 6, were tabulated.

The following are the three computational questions posed in Section 3 along with the conclusions developed in this paper.

1. What type of computational arithmetic should be used? Section 3.1 concludes that unlike arbitrary precision and machine precision, rational arithmetic eliminates all rounding and catastrophic cancelation errors so rational arithmetic in Mathematica was used exclusively in this paper.

2. What arithmetic form should be used to input the test statistic d^+ to the thirteen formulae? When using rational arithmetic to compute the p values, Section 3.2 concludes that the test statistic d^+ must be input as a rational number otherwise the correct p value will not be produced.

3. What is the fastest way to calculate the binomial coefficient in the direct and recursion formulae: (1) the Rational Number Binomial Function (RNBF) method or (2) the Rational Number Iterative Calculation (RNIC) method? Section 3.3 concludes that the RNIC method is slightly faster and more portable than the RNBF method, so this paper used the RNIC method.

The following are the four research questions posed in Section 3 along with the conclusions developed in this paper.

1. What is the best (fastest) way to calculate a bandwidth: (1) binary search or (2) linear search? Preliminary computational experience in Section 5 compared the binary search algorithm to the linear search algorithm and concluded that the linear search algorithm is the fastest.

2. What is the relationship between computation time and sample size n? All the computational experience in Section 6 showed that computational time increases with increasing sample size n. This increase in computational time reflects the fact that the number of terms in every one of the thirteen formulae increases with increasing sample size.

3. Using rational arithmetic, what is the fastest formula? Section 6 compares the computational times for all thirteen formulae and concluded that DwassAltD is the fastest. Section 9 gave the computational times for DwassAltD for samples sizes n up to $n = 50,000$ from which we can conclude that sample size $n = 50,000$ is a practical upper limit for calculating p values using rational arithmetic.

4. How does the computation time change as the accuracy of the test statistic d^+ increases? Section 8 showed that computational time increases as the number of numerator/denominator digits increases in the rational arithmetic representation of the test statistic d^+.

11. Areas of future research

Dvoretzky, Keifer, and Wolfowitz (1956) gave an inequality that showed the two-sided one-sample K-S probability could be approximated by doubling the one-sided one-sample K-S probability. Owen (1962) used this approximation to tabulate two-sided critical values. Future work on the computation of the one-sided one-sample K-S test should investigate other arithmetics that may prove to be more efficient and/or more portable than rational arithmetic. Specifically, arbitrary precision arithmetic implementations, where the internal precision ip is selected so that the final p value will have a resulting precision rp greater than or equal to the user specified desired precision dp, $rp \geq dp$, may produce faster computational times. Once K-S p values can be calculated accurately by arbitrary precision, the error in machine precision arithmetic implementations can be studied. Since the computer statistical software packages listed in Table 1 use machine precision arithmetic to generate p values, it is important to devise machine precision techniques to calculate K-S p values whose accuracy are known. Another area of future research is to evaluate the error in calculating the one-sided one-sample K-S p values using the limiting distribution and other approximations.

Finally, it is always possible that clever implementations of some of the formulae may change their relative efficiency and DwassAltD may not be the fastest. For example, arbitrary precision implementations may considerably change the relative efficiencies of the thirteen formulae.

Acknowledgments

We gratefully acknowledge the financial support the Division of Research and Graduate Studies at Kent State University. The authors are grateful to the Editor and two anonymous referees for their insightful comments and recommendations. Incorporating their suggestions improved the quality of the article.

References

Birnbaum ZW, Tingey FH (1951). "One-sided Confidence Contours for Probability Functions." *Annals of Mathematical Statistics*, **22**(4), 592–596.

Brown JR, Harvey ME (2005). "Comparing Different Formulae to Evaluate the Two-Sided One-sample Kolmogorov-Smirmov Cumulative Sampling Distribution Using Rational Arithmetic." Kent State University Working Paper, August 2005.

Brown JR, Harvey ME (2006). "Binomial Coefficient Series for Rational Arithmetic Mathematica Functions to Evaluate the One-sided One-sample K-S Cumulative Sampling Distribution." Kent State University Working Paper, June 2006.

Conover WJ (1972). "A Kolmogorov Goodness-of-Fit Test for Discontinuous Distributions." *Journal of the American Statistical Association*, **67**(339), 591–596.

Conover WJ (1999). *Practical Nonparametric Statistics*. John Wiley & Sons, New York, third edition.

Daniel WW (1990). *Applied Nonparametric Statistics*. PWS-KENT Publishing Company, Boston, Massachusetts, second edition.

Daniels HE (1945). "The Statistical Theory of the Strength of Bundles of Threads, I." *Proceedings of the Royal Statistical Society A*, **183**, 405–435.

Drew JH, Glen AG, Leemis LM (2000). "Computing the Cumulative Distribution Function of the Kolmogorov-Smirnov Statistic." *Computational Statistics & Data Analysis*, **34**(1), 1–15.

Durbin J (1973). *Distribution Theory for Tests Based on the Sample Distribution Function*. Society for Industrial and Applied Mathematics, Philadelphia.

Dvoretzky A, Keifer J, Wolfowitz J (1956). "Asymptotic Minimax Character of the Sample Distribution Functions and of the Classical Multinomial Estimator." *Annals of Mathematical Statistics*, **27**(3), 642–669.

Dwass M (1959). "The Distribution of the Generalized D_N^+." *Annals of Mathematical Statistics*, **29**(4), 1024–1028.

Feller W (1948). "On the Kolmogorov-Smirnov Limit Theorems for Empirical Distributions." *Annals of Mathematical Statistics*, **19**(2), 177–189.

Gibbons JD, Chakraborti S (2003). *Nonparametric Statistical Inference*. Marcel Dekker, New York, fourth edition.

Kolmogorov A (1933). "Sulla Determinazione Empirica di una Legge di Distribuzione." *Giornala dell'Istitoto Italiano degli Attuari*, **4**, 83–91.

Kotelnikov VF, Chmaladze EV (1983). "On Computing the Probability of an Empirical Process Not Crossing a Curvilinear Boundary." *Theory of Probability and Its Application*, **27**(3), 640–648.

Maag UR, Dicaire G (1971). "On Kolmogorov-Smirnov Type One-sample Statistics." *Biometrika*, **58**(3), 653–656.

Marsaglia G, Tsang WW, Wang J (2003). "Evaluating Kolmogorov's Distribution." *Journal of Statistical Software*, **8**(18), 1–4. URL http://www.jstatsoft.org/v18/i08/.

Massey FJ (1950). "A Note on the Estimation of a Distribution Function By Confidence Limits." *Annals of Mathematical Statistics*, **21**(1), 116–119.

Massey FJ (1951). "The Kolmogorov-Smirnov Test for Goodness of Fit." *Journal of the American Statistical Association*, **46**(1), 68–78.

Noe M (1972). "The Calculation of Distributions of Two-Sided Kolmogorov-Smirnov Type Statistics." *Annals of Mathematical Statistics*, **43**(1), 58–64.

Noe M, Vandewiele G (1968). "The Calculation of Distributions of Kolmogorov-Smirmov Type Statistics Including a Table of Significance Points for a Particular Case." *Annals of Mathematical Statistics*, **39**(1), 233–241.

Owen DB (1962). *Handbook of Statistical Tables*. Addison-Wesley, Reading, Massachusetts.

Press WH, Teukolsky SA, Vetterling WT, Flannery BP (1992). *Numerical Recipes in C*. Cambridge University Press, New York, second edition.

R Development Core Team (2006). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Ruben H, Gambino J (1982). "The Exact Distribution of Kolmogorov's Statistic D_n for $n \leq 10$." *Annals of the Institute Statistical Mathematics*, **34**, 167–173.

Shorack GR, Wellner JA (1986). *Empirical Processes with Applications to Statistics*. John Wiley & Sons, New York.

Smirnov NV (1944). "Approximate Laws of Distribution of Random Variables from Empirical Data." *Uspekhi Matematicheskikh Nauk*, **10**, 179–206.

Smirnov NV (1948). "Table for Estimating the Goodness of Fit of Empirical Distributions." *Annals of Mathematical Statistics*, **19**, 279–281.

Sprent P, Smeeton NC (2001). *Applied Nonparametric Statistical Methods*. Chapman & Hall/CRC, New York.

SPSS Inc (2006). *SPSS 15.0 Command Syntax Reference*. SPSS Inc., Chicago, IL. In file `spssbase.pdf`, URL http://www.spss.com/.

StatSoft, Inc (2006). *Electronic Statistics Textbook*. StatSoft, Inc., Tulsa, OK. URL http://www.statsoft.com/textbook/glosi.html.

Steck GP (1969). "The Smirnov Two Sample Tests as Rank Tests." *Annals of Mathematical Statistics*, **40**(4), 1449–1466.

Stephens M (1970). "'Use of Kolmogorov-Smirnov, Cramér-von Mises and Related Statistics Without Extensive Tables." *Journal of the Royal Statistical Society B*, **32**(1), 115–122.

Visual Numerics (2006). **IMSL** *C Numerical Library: C Stat Library User's Guide*. Visual Numerics, Inc., San Ramon, CA, 6 edition. URL http://www.vni.com/products/imsl/.

Wolfram S (2003). *The Mathematica Book*. Wolfram Media, fifth edition.

Affiliation:

J. Randall Brown
Department of Management & Information Systems
Graduate School of Management
Kent State University
Kent, OH 44240, United States of America
E-mail: jbrown3@kent.edu

Milton E. Harvey
Department of Geography
Kent State University
Kent, OH 44240, United States of America
E-mail: mharvey@kent.edu

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 19, Issue 6 *Submitted:* 2006-06-08
June 2007 *Accepted:* 2007-03-17